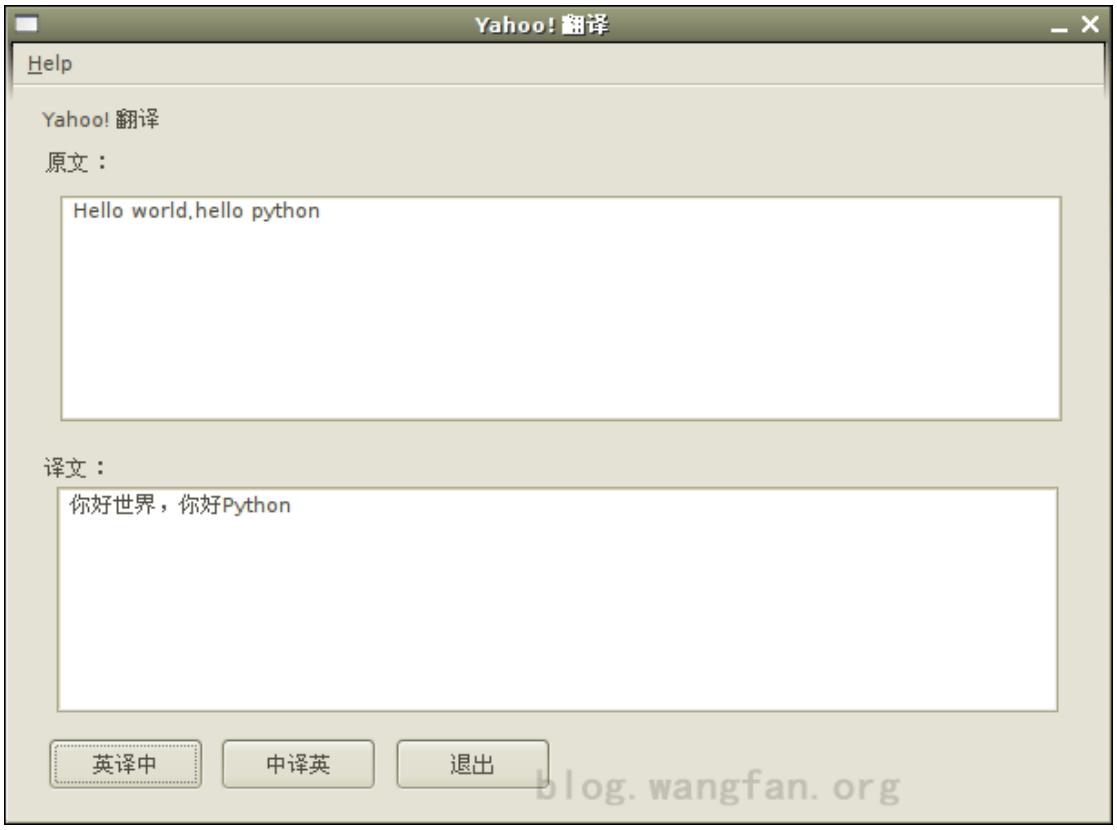
用 python+pygtk+glade3 实现 Yahoo 翻译桌面版
2009-03-15 2200
#转这个代码，方便学习 pygtk，对比了 tk 和 wx，发现还是 pygtk 好点。

#转自 httpblog.wangfan.orgindex.phpa=post&bid=17

每次翻译都去 yahoo 的网站不是很方便，所以自己写了一个 Yahoo 翻译桌面版)

整个程序用 python 做为编码语言， pygtk 库实现 gui。为了方便，用 glade3 画了界面。先来看看最终的程序界面吧：



(由于我用不到别的外语，所以功能上只有英汉互译。)

首先，在开始编码之前，分析一下 yahoo 翻译的网页(httpfanyi.cn.yahoo.com)来确定程序的核心逻辑。Yahoo 翻译的网页源码很简单，一目了然，不像 google 翻译的网页有这么多花哨的 ajax，每次翻译都是向 httpfanyi.cn.yahoo.comtranslate_txtPOST 数据，格式为

“ei=UTF-8&fr=&lp=en_zh&trtext=”

英译中对应的 lp 值为 en_zh，中译英对应的 lp 值为 zh_en,trtext 就是原文的内容。

分析好了就可以开始编码了) 贴出程序的完整源码：

#!usrbinenv python
-- coding utf-8 --

```
#导入库文件
import pygtk
import gtk
import gtk.glade
import urllib
import urllib2
import threading
import re
import sys

#翻译功能类，继承于一个线程类。
class trans(threading.Thread)

    #初始化
    def __init__(self)
        threading.Thread.__init__(self, name='trans')
        self.url='httpfanyi.cn.yahoo.comtranslate_txt'
        self.text=None
        self.trans_in=''
        self.trans_out=''
        self.text_out=None
        self.text_in=None

    def getText(self,widget)
        buf=widget.get_buffer()
        b,e=buf.get_bounds()
        return buf.get_text(b,e)

    def setText(self,widget,text='')
        buf=widget.get_buffer()
        buf.set_text(text)
        widget.set_buffer(buf)

    def run(self)
        self.setText(self.text_out,'正在翻译')
        values={'ei''UTF-8',
                        'fr'',
                        'lp''%s_%s' %(self.trans_in,self.trans_out),
                        'trtext''%s' %(self.getText(self.text_in).replace('n','rnr'),)}
        data=urllib.urlencode(values)
        request=urllib2.Request(self.url,data)
        conn=urllib2.urlopen(request)
        res=conn.read()
```

```
            res=re.findall('div id=pd class=pd (.{1,})div',res)
            res=res[0].replace('br','n')
            res=res.replace('dnt dnt','n')
            self.setText(self.text_out,res)

#主界面类
class yahooTrans()

        def __init__(self)
            self.ui_file=sys.path[0]+'ui.glade'
            self.widgetTree=gtk.glade.XML(self.ui_file,'window1')
            dic={on_exit_clickedgtk.main_quit,
                on_window1_destroygtk.main_quit,
                on_to_zh_clickedself.toZh,
                on_to_en_clickedself.toZh}
            self.widgetTree.signal_autoconnect(dic)

        def toZh(self,widget)
            print widget.get_name()
            t=trans()
            t.text_in=self.widgetTree.get_widget('text_in')
            t.text_out=self.widgetTree.get_widget('text_out')
            if widget.get_name()=='to_zh'
                t.trans_in='en'
                t.trans_out='zh'
            else
                t.trans_in='zh'
                t.trans_out='en'
            t.setDaemon(True)
            t.start()

        def main(self)
            gtk.main()

#入口，gtk 主程序循环
if __name__=='__main__'
        gtk.gdk.threads_init()
        app=yahooTrans()
        app.main()
```

可见，程序中用到了多线程：把翻译过程放在另一个线程中执行。这是因为这个是在线翻译程序，点击翻译按钮后的处理过程（网络通信）相对比较漫长。所以必须使 gtk 主循环与处理过程（翻译）异步。如果直接把翻译过程放在按钮消息映射函数（def toZh(self,widget)）中的话，那么程序在读取网络的时候，整个界面会僵死，直至处理完成。

还有，程序界面用的是 glade 生成的 XML 文档(glade 文件)由 pygtk 通过 gtk.glade.XML 动态载入。实现界面描述与逻辑分离。

另外，整个程序还没有异常处理，所以是很脆弱的。只是作为一个 python + pygtk + glade3 应用的例子)